University of São Paulo
Institute of Mathematics and Statistics
Bachelor of Computer Science

Algebraic Algorithm for Maximum Matching

Antonio Marcos Shiro Arnauts Hachisuca

Final Essay

mac 499 — Capstone Project

Supervisor: Prof. Marcel de Carli Silva

São Paulo
2024

Esta seção é opcional e fica numa página separada;
ela pode ser usada para uma dedicatória ou epígrafe.

Agradecimentos

Do. Or do not. There is no try.
— Mestre Yoda

Texto texto. Texto opcional.

Lista de abreviaturas

URL	Localizador Uniforme de Recursos (*Uniform Resource Locator*)
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

Lista de símbolos

List of Figures

List of Tables

List of Programs

Contents

Introduction

Something.

Chapter 1

Preliminaries

The purpose of this chapter is to introduce key concepts related to the maximum matching algorithm. The chapter covers important topics such as the definition of graph maximum matching, the Sherman-Morrison-Woodbury formula, and the Schur complement. These concepts are fundamental for understanding the algorithm's correctness and time complexity.

1.1 Graph theory

Definition 1.1.1 (Graph). *A **graph** G is a pair (V, E) such that*

*(i) V is a finite set, whose elements are called **vertices**;*
*(ii) E is set of unordered pairs of vertices, whose elements are called **edges**;*

The vertex set of a graph G is denoted as V_G or $V(G)$. The edge set of a graph G is denoted as E_G or $E(G)$. If $\{u, v\} \in E(G)$, then u and v are the **ends** of e and e **incides** in both u and v; When the context is clear, $\{u, v\}$ may be abbreviated to uv.

Definition 1.1.2 (Matching). *For a graph $G := (V, E)$, a subset $M \subseteq E$ is a **matching** of G if and only if no two edges in M share an end. A vertex $v \in V$ is M-covered if some edge of M incides in v, and it is said that M covers v; Otherwise, v is M-exposed. A matching M is:*

- ***maximal**, if there is no edge $e \in E \setminus M$ such that $M \cup \{e\}$ is a matching of G;*
- ***maximum**, if for every matching M' of G one has $|M| \geq |M'|$;*
- ***perfect**, if $|V_G| = |M|$, i.e., every vertex of G is M-covered.*

Definition 1.1.3 (Essential edges). *Let $G := (V, E)$ be a graph that has a perfect matching. An edge $e \in E_G$ is **essential** if the graph $(V_G, E_G \setminus \{e\})$ does not have a perfect matching. An edge is **inessential** if it is not essential.*

Now, the following problem can be introduced.

MAXIMUM MATCHING

Given a graph G, find a maximum matching of G.

1.2 Linear algebra

Definition 1.2.1 (Fields).

Definition 1.2.2 (Submatrix). *Let M be a matrix, we say that M' is a* ***submatrix*** *of M if we can obtain M' by removing zero or more rows and/or columns from M.*

Let M be a matrix. For any sets of indices R and C, we write $M_{R,C}$ or $M[R, C]$ to denote the submatrix of M formed by keeping only the rows indexed by R and columns indexed by C. Furthermore, we use $M[R, *]$ or $M_{R,*}$ to represent the submatrix containing all rows indexed by R and all columns of M (resp., $M[*, C]$).

Theorem 1.2.3 (Sherman-Morrison-Woodbury formula). *Let M be a $n \times n$ matrix, U and V be $n \times k$ matrices. Suppose that M is non-singular. Then*

(1) $M + UV^T$ is non-singular if and only if $I + V^T M^{-1} U$ is non-singular;
(2) If $M + UV^T$ is non-singular, then

$$(M + UV^T)^{-1} = M^{-1} - M^{-1}U(I + V^T M^{-1} U)^{-1} V^T M^{-1}.$$

Proof. For (1).

For (2), it suffices to verify that

$$(M + UV^T)(M^{-1} - M^{-1}U(I + V^T M^{-1} U)^{-1} V^T M^{-1}) = I.$$

Let $A := (I + V^T M^{-1} U)$, then

$$\begin{aligned}
&(M + UV^T)(M^{-1} - M^{-1}U(I + V^T M^{-1} U)^{-1} V^T M^{-1}) \\
&\quad = (M + UV^T)(M^{-1} - M^{-1}UA^{-1}V^T M^{-1}) \\
&\quad = (I - UA^{-1}V^T M^{-1}) + (UV^T M^{-1} - UV^T M^{-1} U A^{-1} V^T M^{-1}) \\
&\quad = (I + UV^T M^{-1}) - (UA^{-1}V^T M^{-1} + UV^T M^{-1} U A^{-1} V^T M^{-1}) \\
&\quad = (I + UV^T M^{-1}) - U(I + V^T M^{-1} U)(A^{-1} V^T M^{-1}) \\
&\quad = (I + UV^T M^{-1}) - UAA^{-1} V^T M^{-1} \\
&\quad = I + UV^T M^{-1} - UV^T M^{-1} = I.
\end{aligned}$$ □

Corollary 1.2.4 (Corollary 2.1 from Harvey's paper). *Let M be a non-singular square matrix, let $N := M^{-1}$ and let S be a subset of rows from M. Let $\tilde{M}$ be a matrix that which is identical to M except that $\tilde{M}_{S,S} \neq M_{S,S}$ and $\Delta := \tilde{M}_{S,S} - M_{S,S}$. If $\tilde{M}$ is non-singular, then*

$$\tilde{M}^{-1} = N - N_{*,S}(I + \Delta N_{S,S})^{-1} \Delta N_{S,*}.$$

Proof. TODO: amsah - Add proof. □

Definition 1.2.5 (Skew-symmetric matrix)**.** *A matrix M is* ***skew-symmetric*** *if $M = -M^T$.*

Fact 1.2.6 (Inverse of a skew-symmetric matrix)**.** *Let M be a skew-symmetric matrix. If M is non-singular, then M^{-1} is also skew-symmetric.*

1.2.1 Time complexities of Matrix algorithms

Let M be a $n \times n$ matrix. The following matrix algorithms can be computed in time complexity $O(n^{\omega})$:

- *Matrix inversion:* Computing the inverse of M, if it exists.
- *Matrix rank:* Determining the rank of M.

Detailed explanations of these algorithms can be found in Section 5.1.

Chapter 2

Perfect matching

2.1 Tutte Matrix

Definition 2.1.1 (Indeterminates).

Definition 2.1.2 (Tutte Matrix). *Let G be a graph and $n := |V_G|$. For each edge $\{u, v\} \in E_G$ associate an indeterminate $t_{\{u,v\}}$. The Tutte matrix is the $n \times n$ skew-symmetric matrix such that $T_{u,v} = \pm t_{\{u,v\}}$ if $\{u, v\} \in E_G$ and 0, otherwise. The sign is chosen such that T is skew-symmetric. The Tutte matrix of G is denoted as T_G.*

For a Tutte matrix T, its Pfaffian, denoted $Pf(T)$, is a polynomial whose complete mathematical definition involves sophisticated algebraic concepts beyond our current scope. However, two fundamental properties of the Pfaffian are crucial:

1. The Pfaffian $Pf(T)$ equals the number of perfect matchings in the underlying graph of T;
2. There exists a fundamental relationship between the Pfaffian and the determinant of a Tutte matrix: $\det(T) = Pf(T)^2$.

For a comprehensive treatment of this remarkable polynomial and its properties, we refer the reader to Chapter 1 of Godsil, 1993.

Fact 2.1.3 (Tutte matrix perfect matching condition). *A graph G has a perfect matching iff T_G is non-singular.*

Proof. Direct from the property $\det(T) = Pf(T)^2$. □

While this property of Tutte matrices is powerful, it presents a computational challenge for algorithmic applications. The issue stems from the relationship $\det(T) = Pf(T)^2$, where $Pf(T)$ contains a term for each perfect matching in graph (G). Since a graph may contain exponentially many perfect matchings, direct computation becomes infeasible.

2.1.1 Probabilistic representation of a Tutte Matrix

A breakthrough came from László, 1979, who demonstrated a probabilistic solution: If we replace the non-zero entries of (T) with random values from a sufficiently large field, the matrix's rank is preserved with **high** probability. This provides a computationally feasible approach.

Lemma 2.1.4 (Schwartz-Zippel)**.** ***[Find the reference itself]*** *Should state that if we evaluate this polynomial at a random point in $F_q^{|E|}$, then the evaluation is zero with probabilty at most n/q. That is something non-zero in indeterminates is zero with random values with probabily at most n/q.*

Proof. Reference Schwartz-Zippel lemma. □

According to Lemma 2.1.4, selecting a sufficiently large field significantly reduces the probability of failure. Constructing such a field is straightforward. For any prime number p, the set of integers modulo p, denoted as $\mathbb{Z}_p$, forms a field of size p.

2.2 Naive algorithm

Using Fact 2.1.3, the following algorithm finds a perfect matching in time $O(n^{\omega+2})$.

Program 2.1 NaiveAlgorithm

```
function NaiveAlgorithm($G$)
    for each $e \in E_G$ do
        $G' \leftarrow (V_G, E_G \setminus \{e\})$
        if $T_{G'}$ is singular then    ▷ By Fact 2.1.3, edge e is inessential.
            $G \leftarrow G'$
        end
    end
    return $E_G$    ▷ Only the essential edges remain in G.
end
```

Let $f(n)$ be the running time of NaiveAlgorithm when $|V_G| =: n$. The running time can be expressed as:

$$f(n) = \binom{n}{2} O(n^\omega) = O(n^2)O(n^\omega) = O(n^{\omega+2}).$$

2.3 Rank-two update algorithm

The bottleneck of the previous algorithm is the necessity to recompute the whole matrix inverse after each iteration.

Theorem 2.3.1 (Rank-two update)**.** *Let G be a graph, $N := T_G^{-1}$ and $S \subseteq V_G$ such that $|S| = 2$. Let $\tilde{T}$ be a matrix which is identical to T_G except that $\tilde{T}_{S,S} = 0$. If $\tilde{T}$ is non-singular,* *then*

$$\tilde{T}^{-1} := N + N_{*,S} \cdot \begin{pmatrix} 1/(1+T_{u,v}N_{u,v}) & 0 \\ 0 & 1/(1+T_{v,u}N_{v,u}) \end{pmatrix} \cdot T_{S,S} \cdot N_{S,*}.$$

Proof. TODO: amsah - Prove $(I - T_{S,S}N_{S,S})^{-1} = \begin{pmatrix} 1/(1+T_{\{u,v\}}N_{\{u,v\}}) & 0 \\ 0 & 1/(1+T_{\{v,u\}}N_{\{v,u\}}) \end{pmatrix}$ □

Corollary 2.3.2 (Edge removal condition)**.** *Let G be a graph, T be the Tutte matrix of G and $N := T^{-1}$. An edge ij is essential if and only if $N_{ij} = -1/T_{ij}$.*

Proof. TODO(amsah): equation (3.6) shows this necessity. □

We use Corollary 2.3.2 to quickly decide if an edge is essential. If it's not, we remove it and update the matrix using a rank-2 update. This removes the necessity to recompute the whole inverse in each iteration.

Program 2.2 RANK-2 UPDATE ALGORITHM

```
FUNCTION RANK2UPDATE($S$, $T$, $N$)   ▷ $|S| = 2$
    return $N + N_{*,S} \cdot \begin{pmatrix} 1/(1+T_{u,v}N_{u,v}) & 0 \\ 0 & 1/(1+T_{v,u}N_{v,u}) \end{pmatrix} \cdot T_{S,S} \cdot N_{S,*}$   ▷ Theorem 2.3.1.
end

FUNCTION RANK2ALGORITHM($G$)
    $T \leftarrow T_G$
    $N \leftarrow T^{-1}$
    for each $\{u,v\} \in E_G$ do
        if $N_{\{u,v\}} = -1/T_{\{u,v\}}$ then   ▷ Corollary 2.3.2
            $N \leftarrow$ RANK2UPDATE($\{u,v\}$, $T$, $N$)
            $T_{u,v} \leftarrow 0$
            $T_{v,u} \leftarrow 0$
        end
    end
    return $E_G$   ▷ Only the essential edges remain in $G$.
end
```

First, let $t(n)$ be the running time of RANK2UPDATE when $|V_G| =: n$. Let A be a $n \times m$ matrix and let B be a $m \times o$. In this context, note that $\tilde{N}T_{S,S}$ is a 2×2 matrix. Therefore, $N_{*,S}\tilde{N}T_{S,S}$ is a $n \times 2$ matrix. Consequently,

$$t(n) = O(2n^2) = O(n^2).$$

Let $g(n)$ be the running time of RANK-2 UPDATE ALGORITHM when $n =: |V_G|$. The running time can be expressed as:

$$g(n) = \binom{n}{2} t(n) = \binom{n}{2} O(n^2) = O(n^2)O(n^2) = O(n^4).$$

Chapter 3

Harvey's algorithm

TODO: Introduction.

3.1 Algorithm

The main bottleneck in the previous algorithm was the need to update the entire inverse matrix at each step. Harvey's algorithm addresses this limitation by employing a divide-and-conquer strategy combined with lazy updates. After each recursive step, only the necessary portions of the inverse matrix are updated. As a result, Harvey's algorithm has a time complexity of $O(n^{\omega})$.

The algorithm will maintain two matrices: T, the Tutte Matrix of the graph, and N, which is initialized as T^{-1}. Remember that using Tutte matrices implies that the algorithm is **probabilistic**.[1] It relies on two recursive functions: DeleteEdgesCrossing and DeleteEdgesWithin.

3.1.1 DeleteEdgesCrossing

DeleteEdgesCrossing(R, S): receives two disjoint sets of vertices R and S and deletes inessential edges that connect a vertex in R to a vertex in S. The following invariant must be preserved:

- DeleteEdgesCrossing(R, S): initially has $N_{R\cup S,R\cup S} = {T^{-1}}_{R\cup S,R\cup S}$ and this property is restored after each call of DeleteEdgesCrossing(R_i, S_j).

To maintain this invariant the following updates are done.

TODO: explicit that in both update theorems T is a Tutte Matrix.

Theorem 3.1.1 (Update 1)**.** *Let R, S be two disjoint sets of vertices such that $|R| = |S| = 1$. Let $N := T^{-1}$, $r \in R$ and $s \in S$. If $\{r, s\}$ is inessential, let $\tilde{T}$ be the Tutte matrix of G without*

[1] I want to add a reminder, but I want to explain in the previous section

edge $\{r, s\}$, then one has

$$\tilde{T}^{-1}_{R,S} = N_{R,S}(1 - T_{R,S}N_{R,S})/(1 + T_{R,S}N_{R,S})$$

and

$$\tilde{T}^{-1}_{S,R} = N_{S,R}(1 - T_{S,R}N_{S,R})/(1 + T_{S,R}N_{S,R}) = -\tilde{T}^{-1}_{R,S}.$$

Proof. The inverse of a Tutte Matrix is skew-symmetric, thus $N_{s,r} = -N_{r,s}$. Let $V := R \cup S$ By Corollary 1.2.4, one has:

$$\begin{aligned}
T'^{-1}_{V,V} &= N_{V,V} - N_{V,V}(I + \Delta N_{V,V})^{-1}\Delta N_{V,V} \\
&= N_{V,V} - N_{V,V}\left(I + \begin{bmatrix} 0 & -T_{r,s} \\ -T_{s,r} & 0 \end{bmatrix}\begin{bmatrix} 0 & N_{r,s} \\ N_{s,r} & 0 \end{bmatrix}\right)^{-1} \Delta N_{V,V} \\
&= N_{V,V} - N_{V,V}\left(I + \begin{bmatrix} -T_{r,s}N_{s,r} & 0 \\ 0 & -T_{s,r}N_{r,s} \end{bmatrix}\right)^{-1} \Delta N_{V,V} \\
&= N_{V,V} - N_{V,V}\left(I + \begin{bmatrix} T_{r,s}N_{r,s} & 0 \\ 0 & T_{r,s}N_{r,s} \end{bmatrix}\right)^{-1} \Delta N_{V,V} \\
&= N_{V,V} - N_{V,V}\left(\begin{bmatrix} 1 + T_{r,s}N_{r,s} & 0 \\ 0 & 1 + T_{r,s}N_{r,s} \end{bmatrix}\right)^{-1} \Delta N_{V,V} \\
&= N_{V,V} - N_{V,V}\begin{bmatrix} \frac{1}{1+T_{r,s}N_{r,s}} & 0 \\ 0 & \frac{1}{1+T_{r,s}N_{r,s}} \end{bmatrix} \Delta N_{V,V} \\
&= N_{V,V} - N_{V,V}\begin{bmatrix} \frac{1}{1+T_{r,s}N_{r,s}} & 0 \\ 0 & \frac{1}{1+T_{r,s}N_{r,s}} \end{bmatrix}\begin{bmatrix} T_{r,s}N_{r,s} & 0 \\ 0 & T_{r,s}N_{r,s} \end{bmatrix} \\
&= N_{V,V} - N_{V,V}\begin{bmatrix} \frac{T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} & 0 \\ 0 & \frac{T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} \end{bmatrix} \\
&= N_{V,V} - \begin{bmatrix} 0 & N_{r,s} \\ N_{s,r} & 0 \end{bmatrix}\begin{bmatrix} \frac{T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} & 0 \\ 0 & \frac{T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} \end{bmatrix} \\
&= N_{V,V} - \begin{bmatrix} 0 & \frac{N_{r,s}T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} \\ \frac{N_{s,r}T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & N_{r,s} - \frac{N_{r,s}T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} \\ N_{s,r} - \frac{N_{s,r}T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} & 0 \end{bmatrix} = \begin{bmatrix} 0 & N_{r,s} - \frac{N_{r,s}T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} \\ N_{s,r} - \frac{N_{s,r}T_{r,s}N_{r,s}}{1+T_{r,s}N_{r,s}} & 0 \end{bmatrix}
\end{aligned}$$

□

Theorem 3.1.2 (Update 2). *Let R, S be two disjoint set of vertices. Let T' be T after removing some (possibly zero) edges from G that connects vertices from R_i to vertices in S_j. Then, let $N := T^{-1}$ and $\Delta := T' - T$, one has:*

$$T'^{-1}_{R\cup S,R\cup S} = N_{R\cup S,R\cup S} - N_{R\cup S,R_i\cup S_j}(I + \Delta N_{R_i\cup S_j,R_i\cup S_j})^{-1}\Delta N_{R_i\cup S_j,R\cup S}.$$

Proof. Direct from Corollary 1.2.4. Update the whole matrix with 1.2.4 and select only the desired submatrix. □

We have the following algorithm.

Program 3.1 Harvey's algorithm: DELETEEDGESCROSSING

```
FUNCTION DeleteEdgesCrossing($R$, $S$)   ▷ $R$ and $S$ are **disjoint** sets of vertices.
    if $|R| = 0$ or $|S| = 0$ then return $\emptyset$   ▷ There are no edges.

    if $|R| = 1$ and $|S| = 1$ then   ▷ There is at most **one** edge.
        Let $r$ in $R$
        Let $s$ in $S$
        if $T_{r,s} \neq 0$ and $T_{r,s} \neq -1/N_{r,s}$ then   ▷ This edge can be removed.
            $N_{r,s} \leftarrow N_{r,s}(1 - T_{r,s}N_{r,s})/(1 + T_{r,s}N_{r,s})$   ▷ Theorem 3.1.1.
            $N_{s,r} \leftarrow -N_{r,s}$
            $T_{rs}, T_{sr} \leftarrow 0, 0$   ▷ Edge has been removed.
        end
        return
    end

    $RS \leftarrow R \cup S$
    $R_1, R_2 \leftarrow$ divide $R$ in two
    $S_1, S_2 \leftarrow$ divide $S$ in two
    for $i$ in $\{1, 2\}$ do
        for $j$ in $\{1, 2\}$ do
            $T', N' \leftarrow T, N$   ▷ Save current $T$ and $N$ states
            DeleteEdgesCrossing($R_i$, $S_j$)
            $\Delta \leftarrow T_{R_i \cup S_j, R_i \cup S_j} - T'_{R_i \cup S_j, R_i \cup S_j}$
            $N_{RS,RS} \leftarrow N'_{RS,RS} - N'_{RS,R_i \cup S_j}(I + \Delta N'_{R_i \cup S_j, R_i \cup S_j})^{-1} \Delta N'_{R_i \cup S_j, RS}$   ▷
                Theorem 3.1.2.
        end
    end
end
```

Time complexity: Let $f(r, s)$ be the running time for DELETEEDGESCROSSING(R, S) when $|R| = r$ and $|S| = s$. Let $n = r + s$. Clearly, $f(r, s) = O(1)$ for either base case; Otherwise, one has:

$$
\begin{aligned}
f(r, s) &= O(r + s) + O(r) + O(s) + 4(2O(n^2) + T(r/2, s/2) + O(n^{\omega})) \\
&= 8O(n^2) + f(r/2, s/2) + 4O(n^{\omega}) \\
&= f(r/2, s/2) + O(n^{\omega}) = 2O(n^{\omega}) = O(n^{\omega}). \qquad (3.1)
\end{aligned}
$$

3.1.2 DELETEEDGESWITHIN

DELETEEDGESWITHIN: receives a set of vertices S and deletes inessential edges that have both ends in S. The following invariant must be preserved:

- DELETEEDGESWITHIN(S): initially has $N_{S,S} = T^{-1}_{S,S}$ and this property is restored after each call of DeleteEdgesWithin and DeleteEdgesCrossing.

To maintain this invariant the following update is done.

Theorem 3.1.3 (Update 3). *Let $S \subseteq V_G$, T' be T after removing some (possibly zero) edges from G that connects vertices from V to vertices in V. Then, let $N := T^{-1}$ and $\Delta := T' - T$, one has*

$$T'^{-1}_{S,S} = N_{S,S} - N_{S,S_i}(I + \Delta N_{S_i,S_i})^{-1}\Delta N_{S_i,S}.$$

Proof. Direct from Corollary 1.2.4. Update the whole matrix with 1.2.4 and select only the desired submatrix. □

We have the following algorithm.

Program 3.2 Harvey's algorithm: DeleteEdgesWithin

```
function DeleteEdgesWithin($S$)
    if $|S| = 1$ return

    $S_1, S_2 \leftarrow$ divide $S$ in the middle
    for $i$ in $\{1, 2\}$ do
        $T', N' \leftarrow T, N$   ▷ Save current $T$ and $N$ states
        DeleteEdgesWithin($S_i$)
        $\Delta \leftarrow T_{S_i,S_i} - T'_{S_i,S_i}$
        $N_{S,S} \leftarrow N' - N'_{S,S_i}(I + \Delta N'_{S_i,S_i})^{-1}\Delta N'_{S_i,S}$   ▷ Theorem 3.1.3.
    end
    DeleteEdgesCrossing($S_1, S_2$)
end
```

Time complexity: Let $g(n)$ be the running time of DeleteEdgesWithin(S) when $|S| = n$. The base case is direct. Then,

$$g(n) = O(n) + 2(2O(n^2) + g(n/2) + O(n^\omega)) + f(n/2, n/2) \tag{3.2}$$
$$= 4O(n^2) + 2g(n/2) + 2O(n^\omega) + f(n/2, n/2) \tag{3.3}$$
$$= 4O(n^2) + 2g(n/2) + 2O(n^\omega) + O(n^\omega) \quad \text{by 3.1} \tag{3.4}$$
$$= 2g(n/2) + 3O(n^\omega) = O(n^\omega) \tag{3.5}$$

3.1.3 PerfectMatching

PerfectMatching: Receives a graph G and finds a perfect matching, or returns $\emptyset$ if one does not exist. It creates a tutte matrix of G using constructor.[2] Note that calling DeleteEdgesWithin($V(G)$) is equivalent to deleting every non-essential edge from G. Thus, after this call, the graph only has essential edges, i.e., edges from the perfect matching. Consequently, there is the following implementation:

[2] I want to properly define a tutte matrix constructor and such in the previous chapter

Program 3.3 Harvey's algorithm: Perfect Matching

```
function PerfectMatching($G$)
    $T \leftarrow$ TutteMatrix($G$)
    if $T$ is singular then return $\emptyset$  ▷ The graph has no perfect matching
    $N \leftarrow T^{-1}$
    DeleteEdgesWithin($V$)
    $M \leftarrow \emptyset$
    for $uv$ in $E(G)$ do
        if $T_{uv} \neq 0$ then $M \leftarrow M \cup \{uv\}$  ▷ This edge was not removed
    end
    if $2|M| = |V|$  ▷ Matching is perfect
        return $M$
    end
end
```

Time complexity: Let $T(n)$ be the running time of PerfectMatching(G) when $|V| = n$ and $E(n)$ be the expected number of iterations until a perfect matching is found. Then,

$$\begin{aligned} T(n) &= O(n^2) + O(n^\omega) + (O(n^\omega) + g(n) + O(n^2) + O(1)) \\ &= (O(n^\omega) + g(n)) \\ &= O(n^\omega + O(n^\omega)) && \text{by 3.5} \\ &= O(n^\omega) \end{aligned}$$

From Lemma 2.1.4, deciding if an edge can be deleted fails with probability n/q where q is the size of the field. Thus, the algorithm fails with probability $\delta < n^3/q$.

Note that the **corretude** of the algorithm is direct from updates performed.

3.2 Analysis

In this section, Harvey's algorithm will be compared with other versions of perfect matching algorithms: the ones implemented in the previous section.[3] The comparisons are made through random tests together with a verifier that asserts the output is a valid perfect matching.

[3] If possible, desired to have Edmonds-Blossoms

Chapter 4

Extension to Maximum Matching

1. Theorem of Lovasz, the size of a maximum matching is the rank of the matrix;
2. Prove this theorem;
3. Extend graph to have a perfect matching and remove added vertices.

Chapter 5

Implementation

5.1 Matrix implementation

In this section, fundamental concepts related to matrix implementation are presented such as time complexity of various matrix operations, matrix multiplication being as hard to compute as matrix inversion and others. The objective is providing a sufficient understanding of key matrix operations.

5.1.1 Definitions

Definition 5.1.1 (Schur complement)**.** *Let M be a square matrix of the form*

$$M = \begin{pmatrix} W & X \\ Y & Z \end{pmatrix}$$

where Z is a square matrix; Then, if Z is non-singular, the matrix

$$C = W - XZ^{-1}Y$$

is the Schur complement of Z in M.

5.1.2 Matrix operations

5.1.3 Matrix inverse

An $O(n^{\omega})$ algorithm that computes the matrix inverse can be achieved exclusively using matrix multiplication with complexity $O(n^{\omega})$. The theorem and pseudo-code below demonstrates the process.

Theorem 5.1.2 (Matrix inversions is no harder than matrix multiplication)**.** *If there is an algorithm that computes matrix multiplication in $O(n^{\omega})$, then there is an algorithm that computes matrix inverse in $O(n^{\omega})$.*

Proof.

$$A^{-1} = \begin{pmatrix} B^{-1} + B^{-1}C^TS^{-1}CB^{-1} & -B^{-1}C^TS^{-1} \\ -S^{-1}CB^{-1} & S^{-1} \end{pmatrix}$$

□

Now, the following algorithm can be implemented.

Program 5.1 Matrix: Inverse

Input: A $n \times n$ matrix A where n is a power of two;
Output: A^{-1}.

```
function MatrixInverse($A$)
    if $n = 1$
        if $A_{0,0} = 0$
 ▷ Matrix is singular.
        end
        $N_{0,0} \leftarrow 1/A_{0,0}$
        return $N$
    end
    $\begin{pmatrix} B & C \\ C^T & D \end{pmatrix} \leftarrow A$   ▷ Each submatrix size should be $n/2 \times n/2$.
    $S \leftarrow D - CB^{-1}C^T$   ▷ Schur complement (5.1.1).
 ▷ Save results for $B^{-1}$ and $S^{-1}$.
    return $\begin{pmatrix} B^{-1} + B^{-1}C^TS^{-1}CB^{-1} & -B^{-1}C^TS^{-1} \\ -S^{-1}CB^{-1} & S^{-1} \end{pmatrix}$   ▷ Theorem 5.1.2.
end
```

Complexity

Note that the algorithm above can be optimized to only calculate the recursive inversions a single time by saving them. Suppose it is optimized. Each call, has two recursive calls one for B^{-1} and another for S^{-1}. Suppose $n = \max(n, m)$.

$$T(n) = O(n^2) + T(n/2, m/2) + O(n^2) + 2O(n^\omega) + 2T(n/2) + O(n^\omega) \tag{5.1}$$

5.1.4 Matrix rank

References

[Godsil 1993] C. D. Godsil. *Algebraic Combinatorics.* Chapman & Hall, 1993 (cit. on p. 7).

[László 1979] Lovász László. “On determinants, matchings and random algorithms”. In: vol. 79. Jan. 1979, pp. 565–574 (cit. on p. 8).

Index